Beschreibung

Verfahren und Anordnung zum Überwachen eines Programms

Die Erfindung betrifft ein Verfahren und eine Anordnung zum Überwachen eines Programms.

Eine Überwachung eines Programms erfolgt üblicherweise mittels spezieller Haltepunkte (sog. Brakepoints), an denen ein Programmierer Debuginformationen einsehen und gegebenenfalls auf eine Fehlfunktionalität des Programms rückschließen kann. Diese händische Maßnahme ist bei einem hinreichend großen Programm mit zahlreichen Unterprogrammen äußerst umständlich und zeitraubend. Weiterhin erfolgt eine Auswertung der Debuginformation hauptsächlich offline, also nicht zur Laufzeit des Programms. Ist das Programm Teil eines verteilten Systems ist eine übersichtliche händische Überwachung nicht mehr koordinierbar.

Die **Aufgabe** der Erfindung besteht darin, ein Verfahren und eine Anordnung zur Überwachung eines Programms anzugeben, das bzw. die auch zur Laufzeit des Programms, insbesondere in einem verteilten System, eine effiziente Überwachung ermöglicht. Gerade für eine Vielzahl von Programmen ist eine derartige Überwachung anhand vorliegender Erfindung machbar.

Diese Aufgabe wird gemäß den Merkmalen der unabhängigen Patentansprüche gelöst. Weiterbildungen der Erfindung ergeben sich auch aus den abhängigen Ansprüchen.

Zur Lösung der Aufgabe wird ein Verfahren zur Überwachung eines Programms angegeben, bei dem das Programm um einen Instrumentierungsteil erweitert wird. Der Instrumentierungsteil generiert eine Nachricht und übermittelt diese Nachricht an einen Überwachungsprozeß. Der Überwachungsprozeß leitet eine vorgegebene Aktion ein.

Ein Instrumentierungsteil ist ein Programmteil oder ein (Programm-)Code(fragment), das mit dem zu überwachenden Programm verbunden wird. Diese Verbindung kann dadurch erfolgen, daß der Instrumentierungsteil in das Programm selbst oder in eine mit dem Programm assoziierte Funktion eingebettet wird. Beispielsweise kann das Programm selbst an einigen Stellen instrumentiert werden, deren jede dann an den Überwachungsprozeß bei bestimmten Ereignissen Nachrichten schickt. Gleichzeitig könnte eine Ausgaberoutine des Programms instrumentiert werden, so daß zusätzlich zur normalen (also ohne Instrumentierung) Ausgabe des Programms weitere Ausgaben an den Überwachungsprozeß geleitet werden (Instrumentieren der Ein-/Ausgabeschnittstelle des Programms).

Der Überwachungsprozeß übernimmt vorzugsweise die Koordination der Nachrichten die (ggf. von zahlreichen verteilten) Programmen übersandt werden. Ferner kann der Überwachungsprozeß über verschiedene Aktionen verfügen, die er als Folge der Nachrichten auslösen bzw. einleiten kann:

- So können z.B. die Nachrichten dargestellt werden. Die Darstellung erfolgt vorzugsweise abhängig von der Zeit, z.B. in Form eines Nachrichtenflußdiagramms (engl.: Message Sequence Chart, MSC). Auch Darstellungsformen als Baumstruktur oder in Form von Listen sind denkbar.

- Weiters ist ein Eingriff in den Ablauf des Programms möglich. So könnte der Instrumentierungsteil veranlassen, daß erst auf eine Antwort von dem Überwachungsprozeß zu warten ist, ehe das Programm selbst fortgesetzt wird. Diese Antwort kann von einem Benutzer/Programmierer vorgenommen werden oder automatisch abhängig von bestimmten Vorgaben erfolgen (IF-THEN-ELSE-Konstruktion). Der automatisierte

Ablauf kann als eigenes Unterprogramm, koordiniert durch den Überwachungsprozeß, ablaufen.

- Auch ist es möglich, daß die Aktion einer Steuerungs- oder Regelungsaufgabe entspricht. Ein Beispiel wäre ein Zu- bzw. Abschalten einer Einheit eines technischen Systems, sobald bestimmte Vorgaben erfüllt sind bzw. an den Überwachungsprozeß gemeldet wurden.

Vorteile des hier vorgestellten Verfahrens sind die Möglichkeit der Überwachung des Programms zu seiner Laufzeit. Durch die Instrumentierung ist sichergestellt, daß auch während der Laufzeit bestimmte Ereignisse in Form von Nachrichten an den Überwachungsprozeß gemeldet werden. Dort werden diese Nachrichten gesammelt und geeignet aufbereitet. Insbesondere kann der Überwachungsprozeß anhand einer Filterfunktion sicherstellen, daß nur diejenigen Nachrichten dargestellt oder berücksichtigt werden, die auch tatsächlich den gefilterten Vorgaben entsprechen. Weiterhin ist die Einsatzmöglichkeit in einem verteilten System ein entscheidender Vorteil. Die unübersichtliche Interaktion vieler verschiedener Programme in dem verteilten System macht eine Fehlersuche schwierig. Die Übermittlung von Nachrichten an den Überwachungsprozeß von jedem instrumentierten (auch verteilten) Programm vieler verschiedener Programme, die nicht alle auf derselben Rechnerplattform ablaufen, ermöglicht - nach geeigneter Filterung und/oder Aufbereitung, z.B. anhand eines Nachrichtenflußdiagramms - die Wahrung einer Übersicht, so daß eine Verfolgung bestimmter Aktion in dem verteilten System und die damit verbundene Fehlersuche deutlich erleichtert werden.

Hierbei sei angemerkt, daß oben beschriebenes Programm durchaus auch je ein Teil eines zusammenhängendes Programms oder eine mit dem Programm verknüpfte (assoziierte) Funktion sein kann.

Besonders von Vorteil ist es, wenn eine mit dem zu überwachenden Programm assoziierte 'Middleware' instrumentiert wird. In diesem Fall kann zentral eine Funktionalität, derer sich das Programm bedient, instrumentiert werden. Dies ist gerade deshalb von herausragendem Vorteil, weil so bspw. nicht alle Ein-/Ausgaberoutinen in dem Programm instrumentiert werden müssen, sondern statt dessen die eine zugehörige Routine in der Middleware instrumentiert wird. Dies entspricht gerade der Instrumentierung aller Ein-/Ausgaberoutinen in dem zu überwachenden Programm. Die tatsächliche Erweiterung der einen Stelle in der Middleware ist äußerst ökonomisch, da nicht das ganze zu überwachende Programm nach besagten Routinen durchsucht werden muß. Die Änderung der Instrumentierung kann zentral an einer Stelle für alle betroffenen Routinen erfolgen (Kapselung der Schnittstelle). Ferner ist diese Art der Instrumentierung sehr flexibel und erweiterbar. Ändert sich das zu überwachende Programm, ist die Instrumentierung für das nächste zu überwachende Programm hinsichtlich der betroffenen Routinen (hier als Beispiel die Ein-/Ausgaberoutinen) bereits fertiggestellt.

Insbesondere werden in einem verteilten System, also in einem System, das mehrere Rechner aufweist, die über ein Netzwerk miteinander verbunden sind, die folgenden Mechanismen in dem Programm überwacht:

- Nachrichtenübermittlung (engl.: message passing, task to task):
  Ein Empfänger (Programm oder Prozeß) wartet auf eine Nachricht. Er kann erst weitermachen, wenn die Nachricht von einem Sender übermittelt worden ist.

- Remote Procedure Call (RPC):
  Beispielsweise versucht ein Prozeß auf einem anderen Rechner ("remote") ein Programm zu starten. Der

Prozeß wartet solange, bis auf dem "remote"-Rechner die Bearbeitung beendet ist. Im Normalfall erhält der Prozeß eine Benachrichtung über Erfolg oder Mißerfolg der Bearbeitung.

- Kontrollfluß (engl.: task join):
  Ein Prozeß wird einem anderen Prozeß hinzugefügt. Der hinzugefügte Prozeß ist von da an nicht länger existent.

Liegt ein verteiltes System vor, das aus mehreren Rechnern besteht, so ist jedes System für sich eigenständig und verfügt z.B. über eine eigene Systemuhr. Um eine semantische Korrektheit sicherzustellen ist es wichtig, Plausibilität, z.B. den Zeitstempel der jeweiligen Nachrichten, zu überprüfen. Jede verschickte Nachricht erhält insbesondere einen Zeitstempel, der zur Synchronisation einsetzbar ist. So kann es vorkommen, daß die Übermittlung einer Nachricht eine bestimmte Zeitdauer benötigt und bei der Darstellung in dem Überwachungsprozeß bei mehreren Nachrichten die zeitliche Abfolge geklärt werden muß. Gehen die Systemuhren der einzelnen Systeme nicht gleich (wovon in der Regel auszugehen ist), so wird von dem Überwachungsprozeß die richtige zeitliche Abfolge der gesendeten Nachrichten bestimmt. Dies geschieht insbesondere mittels einer Heuristik, die überprüft, ob eine Annahme korrekt ist oder nicht. Ist dies der Fall, so wird die zuletzt ermittelte zeitliche Abfolge übernommen, im anderen Fall wird eine nächste Annahme überprüft. Die Überprüfung basiert auf einer Semantik der Nachrichten: So kann eine Antwort nicht vor einer Anfrage eingehen. Dies läßt Rückschlüsse auf die Verhältnisse der beiden involvierten Systemuhren zu. Es wird ein zeitlicher Offset bestimmt, der eine Plausibilität der Systemuhren erfüllt. Der Offset kann angepaßt werden, sobald eine weitere Systemuhr mit einem Bezug zu mindestens einer der beiden bestimmten Systemuhren berücksichtigt werden muß.

Das oben beschriebene Verfahren ist bevorzugt einsetzbar zum Testen, Steuern und Warten des Programms bzw. eines mit dem Programm assoziierten technischen Systems.

Auch wird zur Lösung der Aufgabe eine Anordnung zum Überwachen eines Programms angegeben, bei der eine Prozessoreinheit vorgesehen ist, die derart eingerichtet ist, daß

a) das Programm um einen Instrumentierungsteil erweiterbar ist;
b) der Instrumentierungsteil eine Nachricht generiert und an einen Überwachungsprozeß übermittelt;
c) der Überwachungsprozeß eine Aktion auslöst.

Diese Anordnung ist insbesondere geeignet zur Durchführung des erfindungsgemäßen Verfahrens oder einer seiner vorstehend erläuterten Weiterbildungen.

Ausführungsbeispiele der Erfindung werden nachfolgend anhand der Zeichnung dargestellt und erläutert.

Es zeigen

Fig.1 ein Blockdiagramm mit logischen Komponenten zum Überwachen eines Programms;

Fig.2 verschiedene Darstellungen von zu überwachenden Mechanismen eines verteilten Systems;

Fig.3 eine Prozessoreinheit.

Fig.1 zeigt ein Blockdiagramm mit logischen Komponenten zum Überwachen eines Programms. Das zu überwachende Programm 101 ist mit einer Filterfunktionalität 102 und einer Instrumentierung 103 versehen. Der oben vielfach erwähnte Überwachungsprozeß umfaßt die Blöcke Ereignisgenerator 104

(engl.: "Event Generator"), Ereignismanager 109 (engl.: "Event Manager") und die verschiedenen Darstellungsmöglichkeiten 115 bis 120 (engl.: "Views").

Der Ereignisgenerator 104 enthält einen Parser 105, zwei Nachrichtenschnittstellen 106 und 107 und eine Datei 108). Der Ereignismanager 109 umfaßt eine Verbindungseinheit 110 (engl.: "Merger"), eine Ereignisliste 111 (engl.: "Trace Event List"), ein Filtereinheit 112 und eine gefilterte Liste 113. Eine Auswahl des jeweiligen Eintrags innerhalb der gefilterten Liste 113 erfolgt über eine Navigationseinheit 114 (engl.: "Stepper").

Die Filterfunktionalität 102 ist vorgesehen zur Selektion bestimmter für die jeweilige Anwendung interessierender Instrumentierungsteile der Instrumentation 103. Das Ergebnis der Instrumentierung, die Nachrichten, werden an eine Kommunikationsschnittstelle "Socket" 107 übermittelt oder in eine Datei 108 geschrieben. Die eingehenden Nachrichten werden in dem Parser 105 aufbereitet und an die Verbindungseinheit 110 übermittelt. Dort werden die vom Parser 105 erhaltenen Einträge der Nachrichten in einer Liste 111 abgelegt. Insbesondere erfolgt hier auch die semantische Überprüfung der den Nachrichten anhaftenden Zeitstempel. Optional erfolgt in Block 112 eine Filterung aller von der Verbindungseinheit 110 in der Liste 111 abgelegten Nachrichten nach bestimmten Vorgaben. Das Ergebnis ist die gefilterte Liste 113, deren Einträge mittels Navigationseinheit 114 auswählbar sind.

Die Navigationseinheit 114 dient als Interaktionsmedium für den Benutzer. Dort können Eingaben erfolgen, die über die Kommunikationsschnittstelle 106 des Ereignisgenerators 104 an den instrumentierten Programmteil 103 weitergeleitet werden. Insbesondere ist durch diese Interaktion ein Zusammenspiel von Programm und Instrumentierung möglich.

Die Interaktion kann durch Eingabe des Benutzers oder alternativ über eine Vorschrift nach dem Schema "immer wenn ... dann ..." erfolgen. Weiterhin kann eine Aktion 121 durchgeführt werden, die ausgehend von dem Überwachungsprozeß zu einer externen Steuerung oder einem sonstigen externen Eingriff führt.

Die gefilterte Liste 113 kann in unterschiedlichen Darstellungen 115 bis 120 aufbereitet werden. Die Daten der gefilterten Liste 113 können dargestellt werden als

- Liste (Block 115),
- hierarchisch sortierte Ansicht (z.B. nach Rechnern oder Struktur von Programmen und Prozessen, Block 116),
- Nachrichtenflußdiagramm (MSC, Block 117), wobei eine vorgegebene Gruppierung berücksichtigt werden kann,
- detaillierte Ansicht (zu jeder Nachricht gibt es eine Reihe von Informationen betreffend Sender, Empfänger, Zeitstempel, Nachrichteninhalt, etc.; Block 118),
- Liste der vom Benutzer durchgeführten Eingaben (Block 119),
- Testreport (Block 120).

Aus **Fig.2** gehen verschiedene Darstellungen von zu überwachenden Mechanismen eines verteilten Systems hervor. Fig.2 ist dazu dreigeteilt in die Figuren 2A, 2B und 2C. Die jeweilige horizontale Linie bezeichnet die Zeitachse t.

In Fig.2A ist die Nachrichtenübermittlung dargestellt. Ein Sender 201 setzt zu einem Zeitpunkt t1 eine Nachricht ab, die zu einem Zeitpunkt t2 bei einem Empfänger 202 ankommt. Bereits zu einem Zeitpunkt t3 hat der Empfänger 203 damit begonnen auf diese Nachricht zu warten. Bis zu dem Zeitpunkt t2 hat er gewartet, jetzt setzt der Empfänger 202 die Arbeit fort.

Fig.2B zeigt den Mechanismus "Remote Procedure Call, RPC". Zu einem Zeitpunkt t4 setzt ein Prozeß 203 an einen Adressaten 204 eine Meldung "CLIENT_BEGIN(Call object.method(...)" ab. Diese Nachricht kommt zu einem Zeitpunkt t5 bei dem Adressaten 204 an, der vorzugsweise eine eigener Rechner, fern von dem Rechner auf dem der Prozeß 203 abläuft, ist. Der Adressat 204 veranlaßt zu diesem Zeitpunkt t5 mit dem Befehl "SERVER_BEGIN" die Ausführung des übermittelten Kommandos. Zu einem Zeitpunkt t6 ist die Ausführung bei Adressaten 204 beendet, die Arbeit wird eingestellt ("SERVER_END") und das Ergebnis "res" an den Prozeß 203 zurückgeschickt ("Return res=obj.meth(...)"). Zu einem Zeitpunkt t7 kommt das Ergebnis bei dem Prozeß 203 an, der RPC wird mit dem Befehl "CLIENT_END" beendet. Der Prozeß hat von t4 bis t7 gewartet und setzt jetzt mit dem auf dem Adressaten 204 ermittelten Ergebnis "res" seine Bearbeitung fort.

Ein Beispiel für einen Kontrollfluß ist in Fig.2C dargestellt. Dabei sollen zwei Prozesse miteinander verbunden werden ("Task Join"). Zu einem Zeitpunkt t8 wird von einem Prozeß Task1 205 ein Kommando "JOIN_REQUEST" zu einem Prozeß Task2 206 geschickt. Dort angekommen, bewirkt zu einem Zeitpunkt t9 ein Kommando "JOIN_START", daß bis zu einem Zeitpunkt t10 der Prozeß Task2 206 die Verknüpfung mit dem Prozeß Task1 205 durchführt. Zu dem Zeitpunkt t10 erfolgt mit dem Kommando "JOIN_DONE" eine Rückkehr zu dem Prozeß Task1 205, der zu einem Zeitpunkt t11 mit einem Kommando "JOIN_END" die Arbeit wiederaufnimmt. Der Prozeß Task1 205 hat von t8 bis t11 gewartet, der Prozeß Task2 206 ist nach t10 beendet.

Die Figuren 2A, 2B und 2C veranschaulichen unterschiedliche Mechanismen, die in einem verteilten System beobachtet werden können, um einen Überblick über die Interaktion von verteilt ablaufenden Programmen und Prozessen zu erhalten. Gerade bei der Fehlersuche oder bei Wartungsarbeiten ist es von unschätzbarem Vorteil, an der interprozessualen Schnittstelle Aktionen mitverfolgen zu können.

In **Fig.3** ist eine Prozessoreinheit PRZE dargestellt. Die Prozessoreinheit PRZE umfaßt einen Prozessor CPU, einen Speicher SPE und eine Input/Output-Schnittstelle IOS, die über ein Interface IFC auf unterschiedliche Art und Weise genutzt wird: Über eine Grafikschnittstelle wird eine Ausgabe auf einem Monitor MON sichtbar und/oder auf einem Drucker PRT ausgegeben. Eine Eingabe erfolgt über eine Maus MAS oder eine Tastatur TAST. Auch verfügt die Prozessoreinheit PRZE über einen Datenbus BUS, der die Verbindung von einem Speicher MEM, dem Prozessor CPU und der Input/Output-Schnittstelle IOS gewährleistet. Weiterhin sind an den Datenbus BUS zusätzliche Komponenten anschließbar, z.B. zusätzlicher Speicher, Datenspeicher (Festplatte) oder Scanner.

Patentansprüche

1. Verfahren zum Überwachen eines Programms,
   a) bei dem das Programm um einen Instrumentierungsteil erweitert wird;
   b) bei dem der Instrumentierungsteil eine Nachricht generiert und an einen Überwachungsprozeß übermittelt;
   c) bei dem der Überwachungsprozeß eine Aktion auslöst.

2. Verfahren nach Anspruch 1,
   bei dem die Aktion eine der folgenden Möglichkeiten umfaßt:
   d) Darstellung der Nachricht;
   e) Eingriff in den Ablauf des Programms;
   f) Steuerung und/oder Regelung einer mit dem Programm assoziierten Einheit.

3. Verfahren nach Anspruch 1 oder 2,
   bei dem der Instrumentierungsteil nach Übermittlung der Nachricht auf eine Antwort, die von dem Überwachungsprozeß erzeugt wird, wartet.

4. Verfahren nach Anspruch 3,
   bei dem die Antwort nach Eingabe eines Benutzers oder durch einen automatisierten Ablauf erstellt wird.

5. Verfahren nach einem der vorhergehenden Ansprüche,
   bei dem mehrere Nachrichten dargestellt werden als Liste, Baumdiagramm oder als ein Nachrichtenflußdiagramme (Message Sequence Chart, MSC).

6. Verfahren nach einem der vorhergehenden Ansprüche,
   bei dem das Programm ein Teil eines größeren Programms ist.

7. Verfahren nach einem der vorhergehenden Ansprüche, bei dem eine mit dem Programm assoziierte Funktion instrumentiert wird.

8. Verfahren nach einem der vorhergehenden Ansprüche, bei dem eine mit dem Programm assoziierte Middleware instrumentiert wird.

9. Verfahren nach einem der vorhergehenden Ansprüche, bei dem mindestens einer der folgenden Mechanismen überwacht wird:
   g) Remote Procedure Call (RPCs),
   h) Nachrichtenübermittlung,
   i) Kontrollfluß.

10. Verfahren nach einem der vorhergehenden Ansprüche, bei dem mehrere Programm in einem verteilten System oder ein Programm, das über das System verteilt ist, überwacht wird.

11. Verfahren nach Anspruch 10, bei dem eine Überprüfung einer semantischen Korrektheit mittels vorgegebener Heuristiken erfolgt.

12. Verfahren nach einem der vorhergehenden Ansprüche zum Testen, Steuern oder Warten eines technischen Systems.

13. Anordnung zum Überwachen eines Programms, bei der eine Prozessoreinheit vorgesehen ist, die derart eingerichtet ist, daß
   j) das Programm um einen Instrumentierungsteil erweiterbar ist;
   k) der Instrumentierungsteil eine Nachricht generiert und an einen Überwachungsprozeß übermittelt;
   l) der Überwachungsprozeß eine Aktion auslöst.

Zusammenfassung

Verfahren und Anordnung zum Überwachen eines Programms

Es wird ein Verfahren zur Überwachung eines Programms angegeben, bei dem das Programm um einen Instrumentierungsteil erweitert wird. Der Instrumentierungsteil generiert eine Nachricht und übermittelt diese Nachricht an einen Überwachungsprozeß. Der Überwachungsprozeß leitet eine vorgegebene Aktion ein.

FIG 1

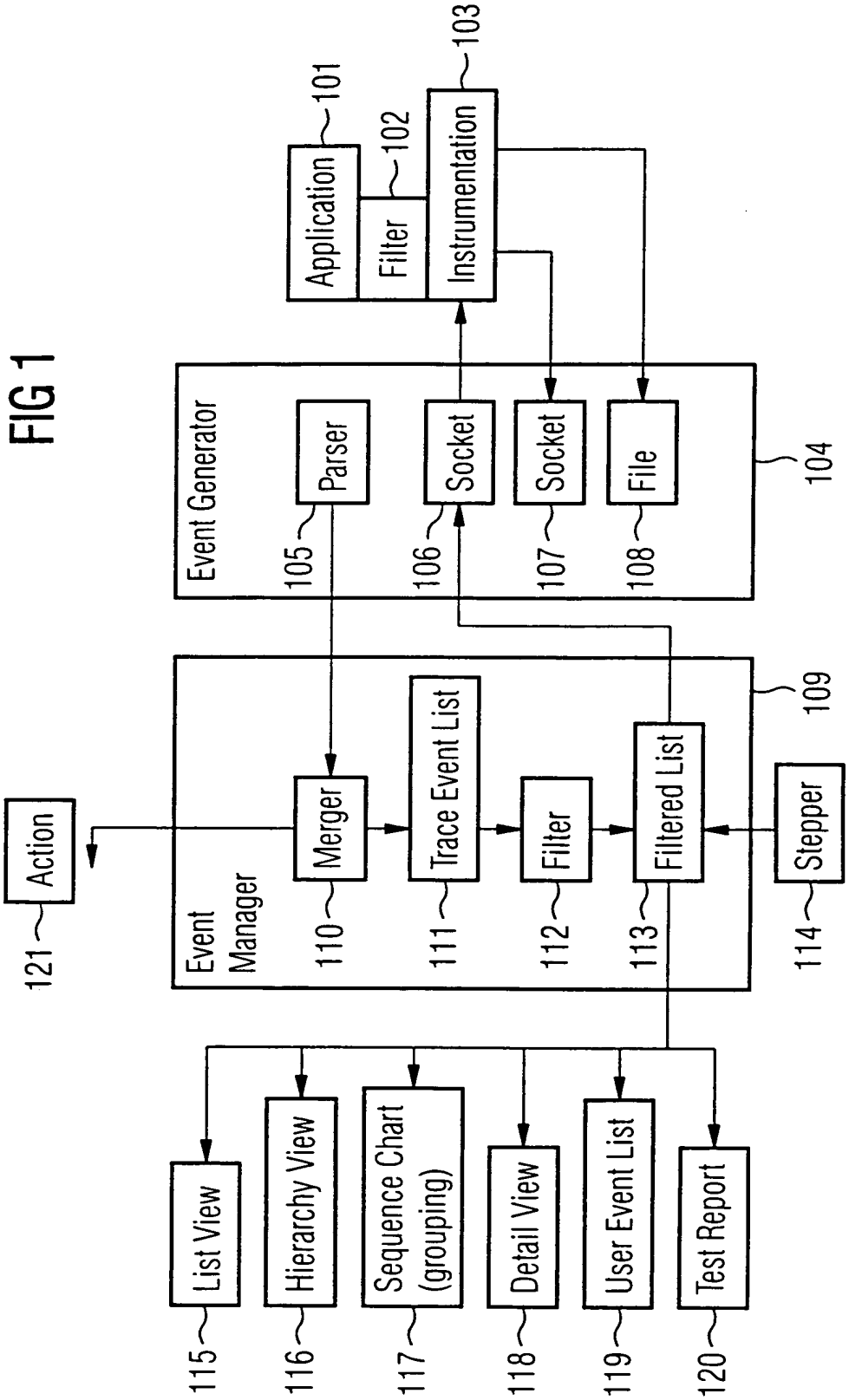

Application
101
Filter
102
Instrumentation
103
Event Generator
Parser
105
Socket
106
Socket
107
File
108
104
Action
121
Event Manager
Merger
110
Trace Event List
111
Filter
112
Filtered List
113
Stepper
114
109
List View
115
Hierarchy View
116
Sequence Chart (grouping)
117
Detail View
118
User Event List
119
Test Report
120

FIG 2

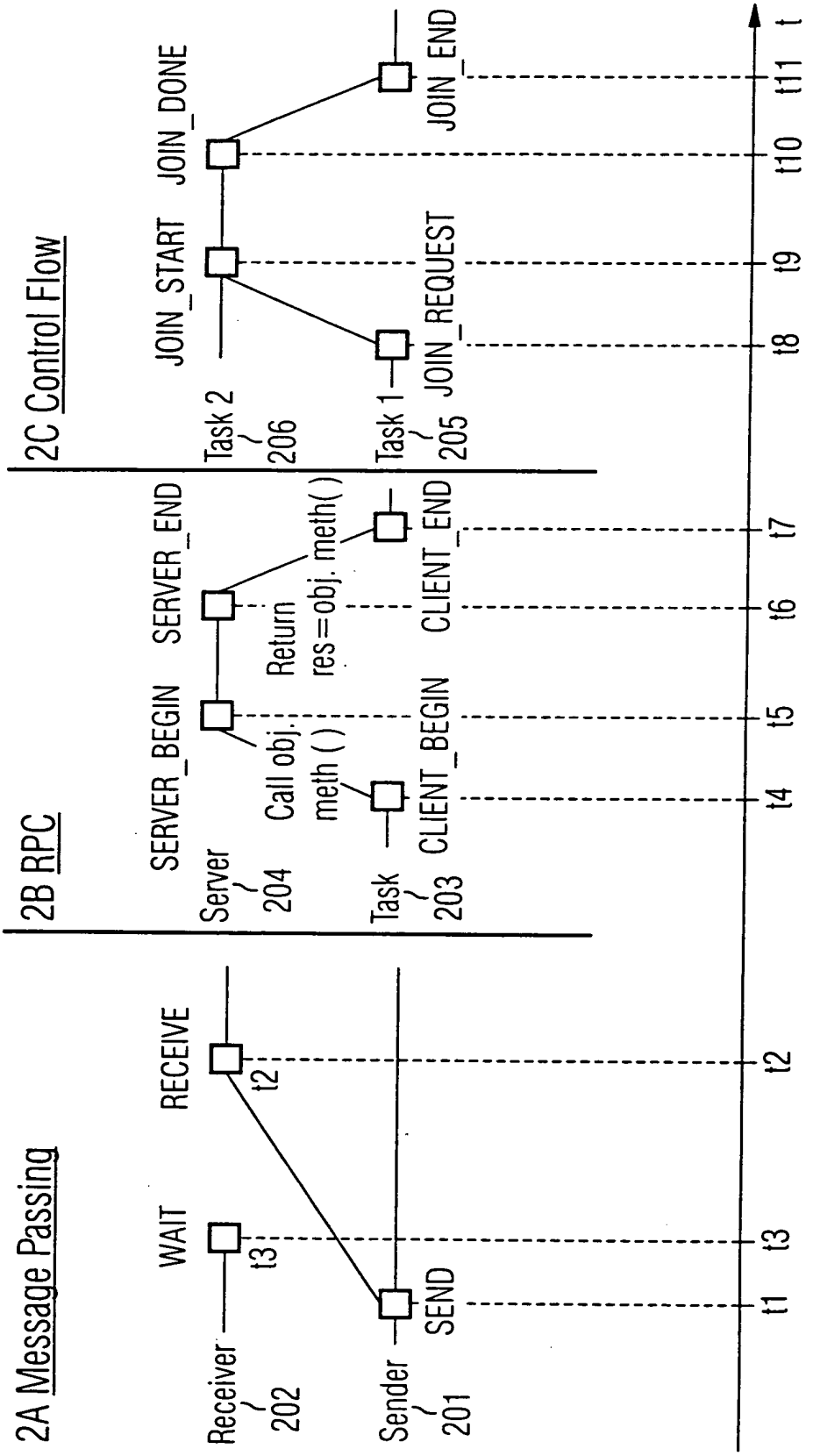
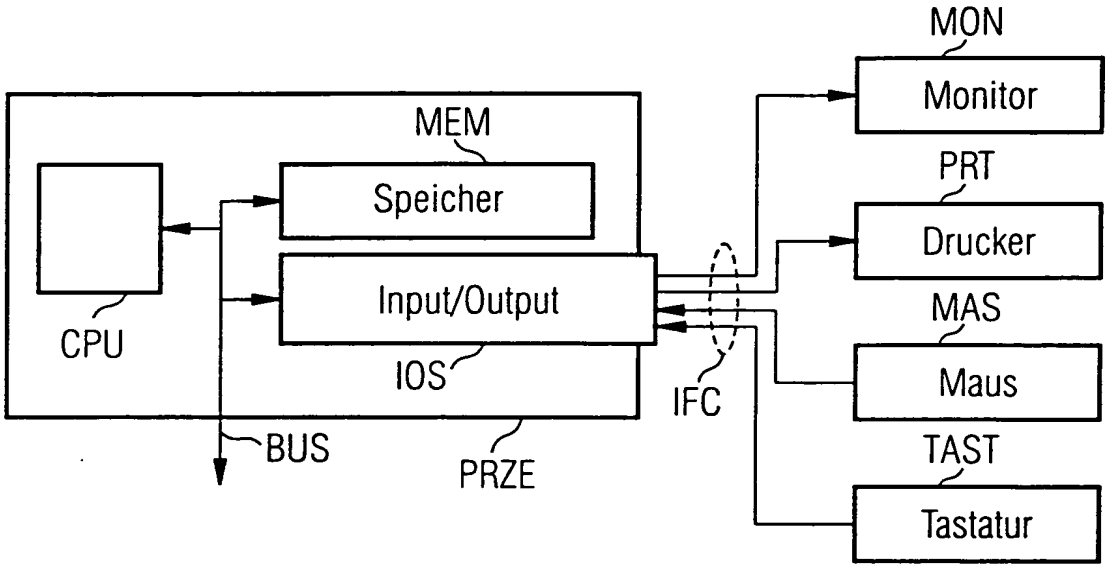

2A Message Passing
Receiver
202
Sender
201
WAIT
RECEIVE
SEND
t1
t2
t3
2B RPC
Server
204
Task
203
SERVER_BEGIN
SERVER_END
CLIENT_BEGIN
CLIENT_END
Call obj. meth ( )
Return res=obj. meth( )
t4
t5
t6
t7
2C Control Flow
Task 2
206
Task 1
205
JOIN_START
JOIN_DONE
JOIN_REQUEST
JOIN_END
t8
t9
t10
t11
t

FIG 3


MON
Monitor
MEM
Speicher
PRT
Drucker
Input/Output
MAS
Maus
CPU
IOS
IFC
BUS
PRZE
TAST
Tastatur